



The 5 Success Factors to deploy Yocto for “Production Grade” Embedded/IoT Devices

Adrien Leravat – Senior Software Architect



Software Partner

Design, Build and Run Edge to Cloud Platforms



19 years in embedded and IoT software

5 offices in US, Germany, UK and France

IoT Services by Avnet Company



Cloud Partners



Tech Partners

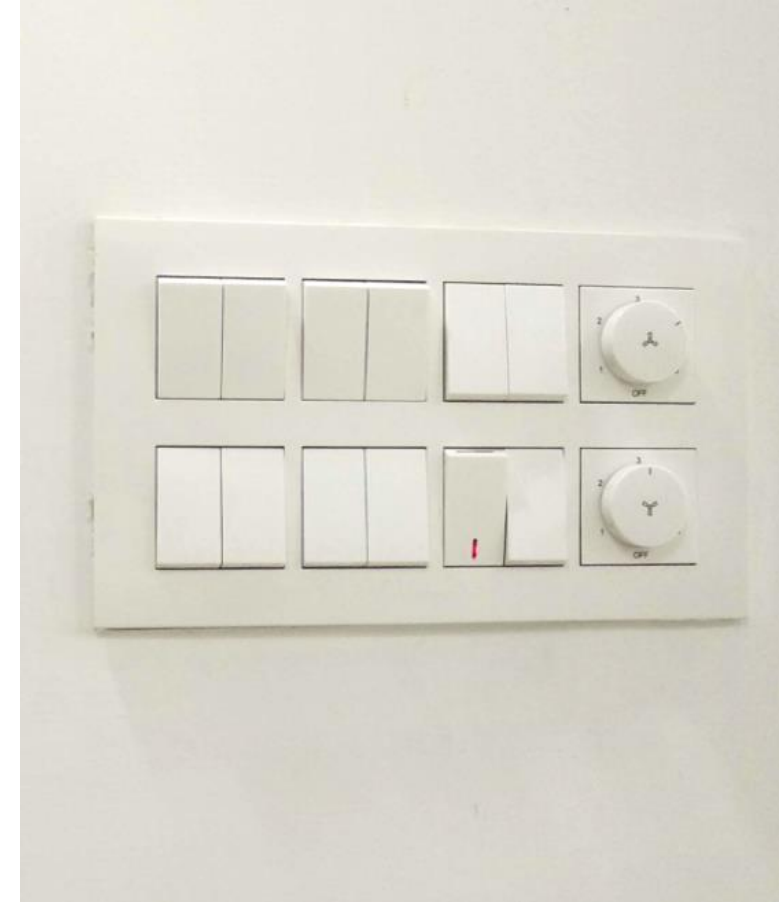


Production grade?



“Works”

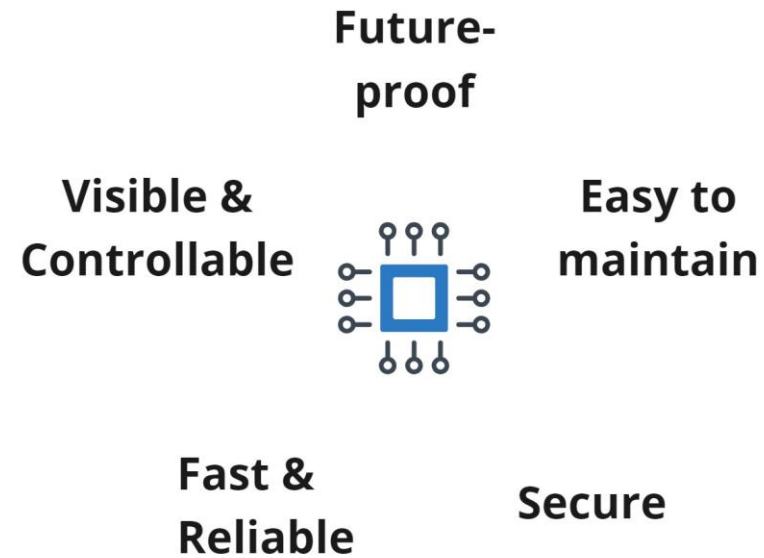
VS



Works!

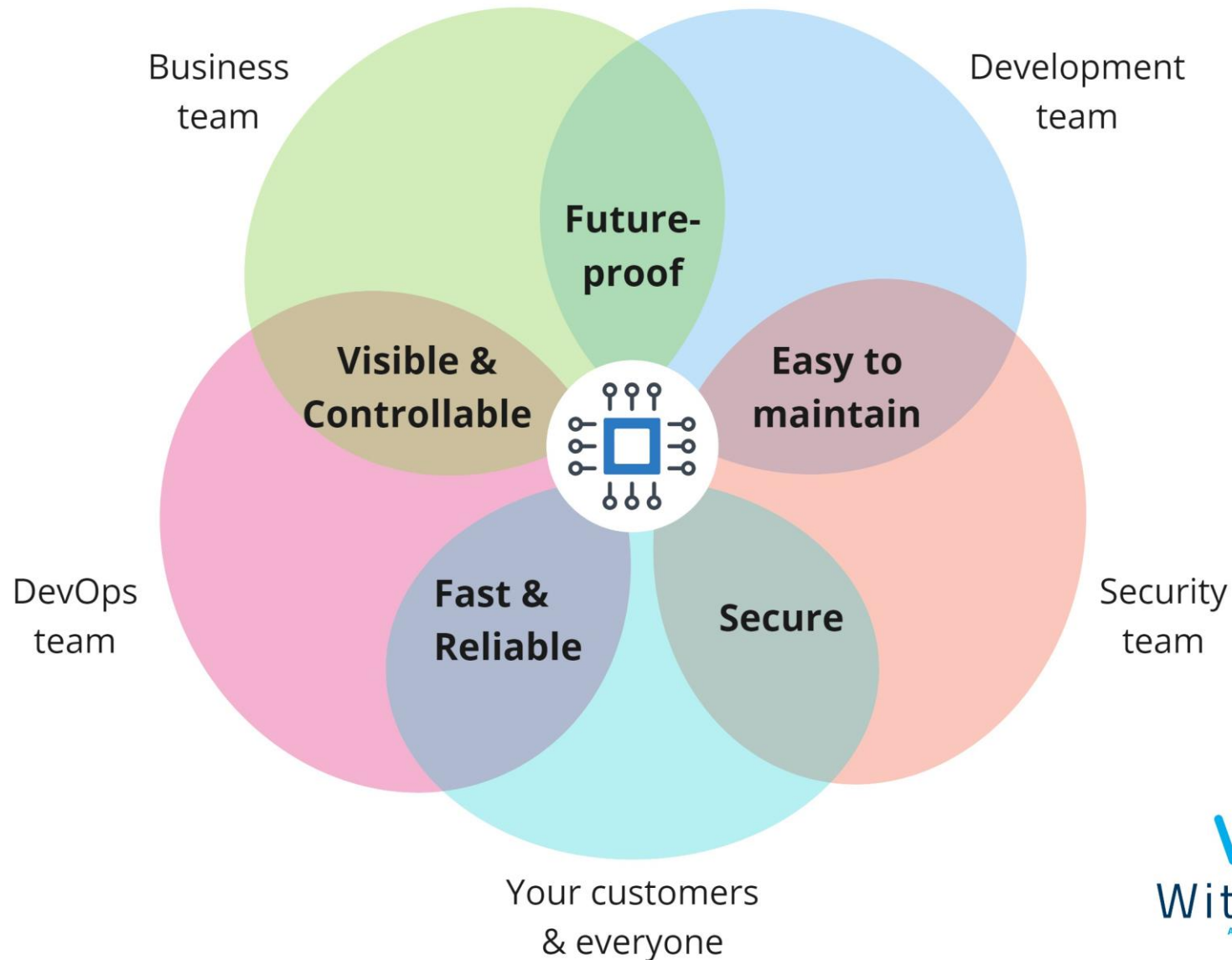
The 5 success factors

Production-ready IoT device




The 5 success factors

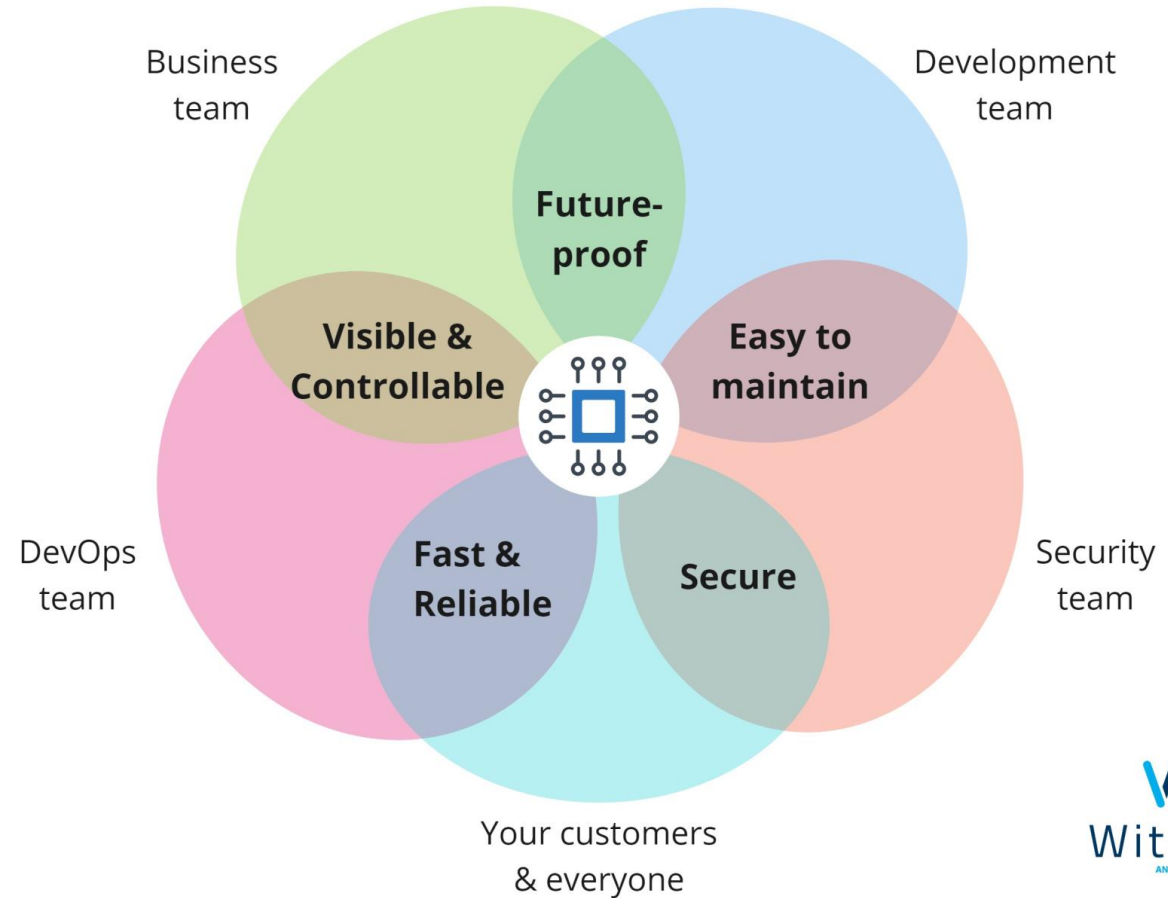
Production-ready IoT device



The 5 key factors

Production-ready IoT device

- Checklist for these 5 area
- Look for checkboxes 
- Some are “Nice to have”
- Aims to provide a 360 view, to be adapted on a case-by-case basis



Devices should be...

1. Easy to Maintain



Photo by [Markus Spiske](#) on [Unsplash](#)

EASY TO MAINTAIN

Easy to work with

❑ Provide dev. environment

- Git, python, repo, ...
- Container and companion scripts
- Prefer prebuilt container image
 - Stable environment and package versions

❑ Document the main dev. workflow

- Essential to newcomers and to share knowledge within the team
- Tough learning curve otherwise ☑
- Crunch time = no time for doc but help wanted!

❑ Shrink build times

- Share downloads and sstate (NFS, SSHFS, ...)
 - site.conf:
 DOWNLOAD_DIR =
 SSTATE_DIR =
 SSTATE_MIRRORS =
- Ask for a beefier machine 😊

❑ Provide an SDK for app. devs

- So much faster to build... and less troubles
- QEMU for prototyping
- Alternatively: build apps as package for quick deployment

Plus a couple of other best practices in the last part...

EASY TO MAINTAIN

Versioned, reproducible builds

❑ Version your OS and pin everything

- Version code, layers, and configuration
 - repo, git submodules, ...
- Pin meta-layers tags or commit
- Tag your OS
- Update “os-release”
 - Version, type of build, ...

❑ Automate Yocto builds

- Nightlies: relatively easy, cache downloads and sstate
- Automate from cloud, build on premise
 - GitLab runners, Azure DevOps self-hosted agents
- Pull request validation: highly valuable
 - Trigger build from another repo, override package or layer
 - SRCREV_pn-\$PACKAGE_NAME =

❑ (Nice to have) Archive release build environment

- Yocto: Reproducible Builds affiliated ✓
- Your setup 1 year from now: likely KO 😞
 - Main culprit: external tools
- Archive source, tools, environment
 - Downloads
 - VM or **prebuilt** image
- Annotate your release with its build environment
- Keep the build manifest: layers & packages version hash

Devices should be...

2. Secure



Photo by [Julia Kamm](#) on [Unsplash](#)

SECURE

OS Security features

☐ Minimize attack surface

- Define prod and prod-secure images
- No dev/debug tools, or unnecessary packages
 - Image from scratch, core-image-minimal
 - Disable recommended packages
- No root login, proper users, firewall
- Disable serial, JTAG, USB (or exceptions)
- Secure protocols, meta-security

☐ Use Secure boot

- BootROM, SSB, U-Boot, Kernel/initramfs/dtb
- Rootfs if readonly
 - dm-verity
- Use key hashes, 1-2 backup secure boot keys, and support revocation with efuses

- ... And test it!

☐ Provide a secure secret store

- TPM, Secure Element, TrustZone-based
- Device keys, credentials, secrets

☐ (Nice to have) Encrypt disks, prevent writes

- Encrypt partitions, LUKS, dm-crypt, and secret store
- Read-only filesystem (SquashFS), or mounted read-only

SECURE

Other features

❑ Apps: Least privilege principle

- MAC: SELinux, AppArmor
- Containers, AWS Greengrass, Azure IoT Edge

❑ Monitor and address vulnerabilities

- INHERIT += “cve-check”
- Shows unpatched vulnerabilities, based on the package version
- Time consuming
 - meta-timesys provides a few tools
 - Commercial maintenance solutions available (including Witekio)
- **No package = no exploit**

❑ (Nice to have) Run confidential code in secure environment

- TrustZone, OP-TEE

❑ (Nice to have) Consider standards and regulations

- Likely to become more important
- Europe: ETSI EN 303 645
- US: NIST 8259A

SECURE

Device identity

❑ Automate on-device identity provisioning

- Unique x.509 certificates
- Signed from an intermediate certificate, or pre-provisioned on-chip in a secure element (e.g. TO136)

❑ Automate device provisioning in the cloud

- Azure DPS, AWS
- Option A. Authorize an intermediate cert
- Option B. Pre-provisioned secure elements (done for you)
 - Done during chip manufacturing, from a secure software factory (TO136)
- Option C. Automate and secure

❑ Support rolling device certs, and updating root CAs

- Software update or otherwise, and test carefully
- Root CAs too!



Devices should be...

3. Fast and reliable



Photo by [SpaceX](#) on [Unsplash](#)

FAST & RELIABLE

Boot and runtime

Optimize only as required, measure and profile

❑ Fast (enough) Boot time

- Start from a minimal image
- Compile for size, link statically, strip binaries, use to musl or uClibc
- Postpone drivers and services
- Btrfs, squashfs
- ... or just dump a small logo/animation from the U-Boot!

❑ Fast (enough) & Responsive UX

- Compile for speed
- Leverage cores, CPU instructions, priority
- 2D, 3D, and video hardware acceleration
- Crypto hardware acceleration
- Accelerated libs: GUI, AI inference, ...



FAST & RELIABLE

Automated tests, staging, deployment

Yocto and test automation

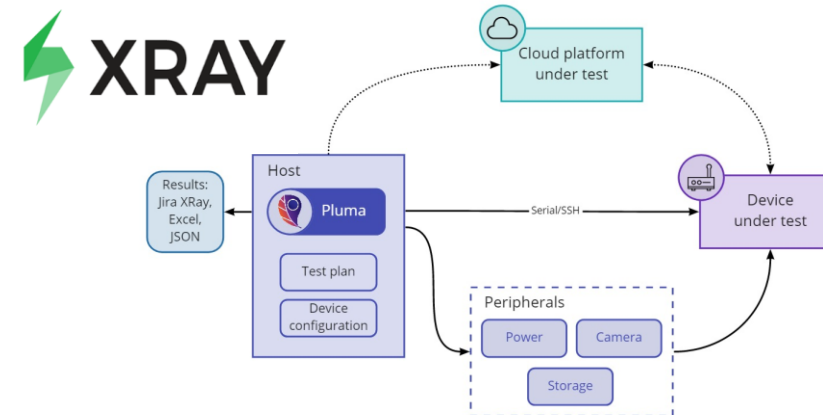
- Based on AutoBuilder2, helper, and Buildbot
 - Not necessarily a good fit for you

□ Automate on-device tests

- Ptest, ptest-runner
 - `DISTRO_FEATURES:append = " ptest"`
`EXTRA_IMAGE_FEATURES += "ptest-pkgs"`
- LTP (Linux Test Project)
 - `IMAGE_INSTALL:append = " ltp"`
- And device-specific tests
- Automate with Labgrid, Pluma, Fuego, Lava, Buildbot, KernelCI
- Integrate with GitLab runners

□ (Nice to have) Automate tagging and deployment

- Manual release “trigger”, automated release
- More consistent, less errors
- Can tag, archive build environment, ...
- Push to your OTA update backend and/or package host



Devices should be...

4.1 Visible (Observable)

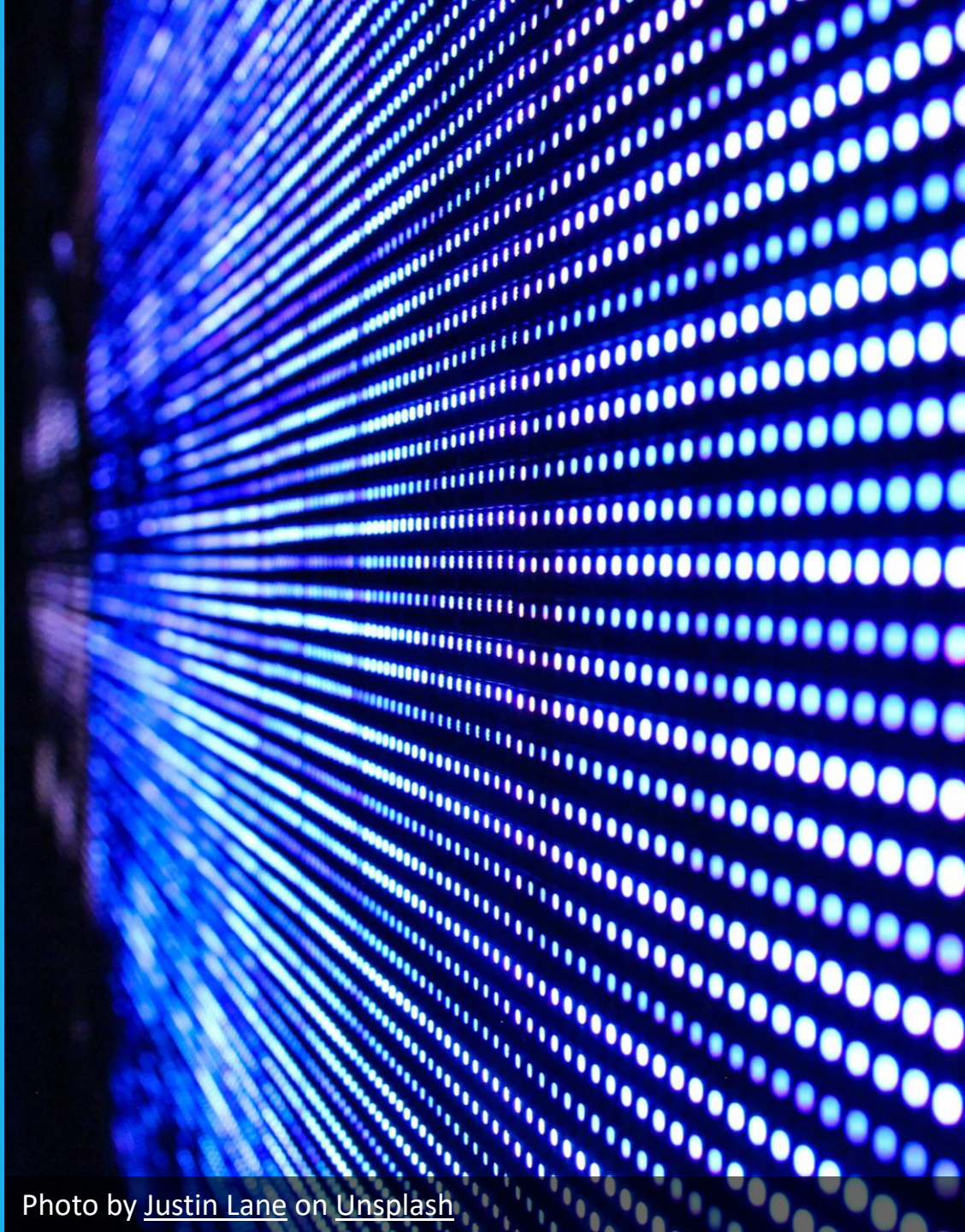


Photo by [Justin Lane](#) on [Unsplash](#)

VISIBLE (OBSERVABLE)

Device state and issues

❑ Expose device state

- Device twins (Azure), device shadow (AWS)
 - JSON with a device and cloud section
- Think of helpful states to share
 - Battery, Boot slot, storage health, secure boot on/off
- Periodically store in a file or twin directly
- Status monitoring: AWS IoT Core API

❑ Send logs and usage info

- journald, syslog
- Connectors: syslog-ng (meta-oe), Filebeats
- Send to ELK, Azure/AWS IoT, Monitor, LogWatch
- Essential for SREs and Security teams (SIEM)
- Bonus: Google Analytics for marketing KPI and usage

❑ (Nice to have) Provide Ops and BI dashboard

- Essential for larger fleets
- From device status, logs, messages
- Thingsboard, IoT Connect, AWS IoT Device Manager, AWS IoT SiteWise, (Azure) Power BI

❑ (Nice to have) Support (future) big data scenarios

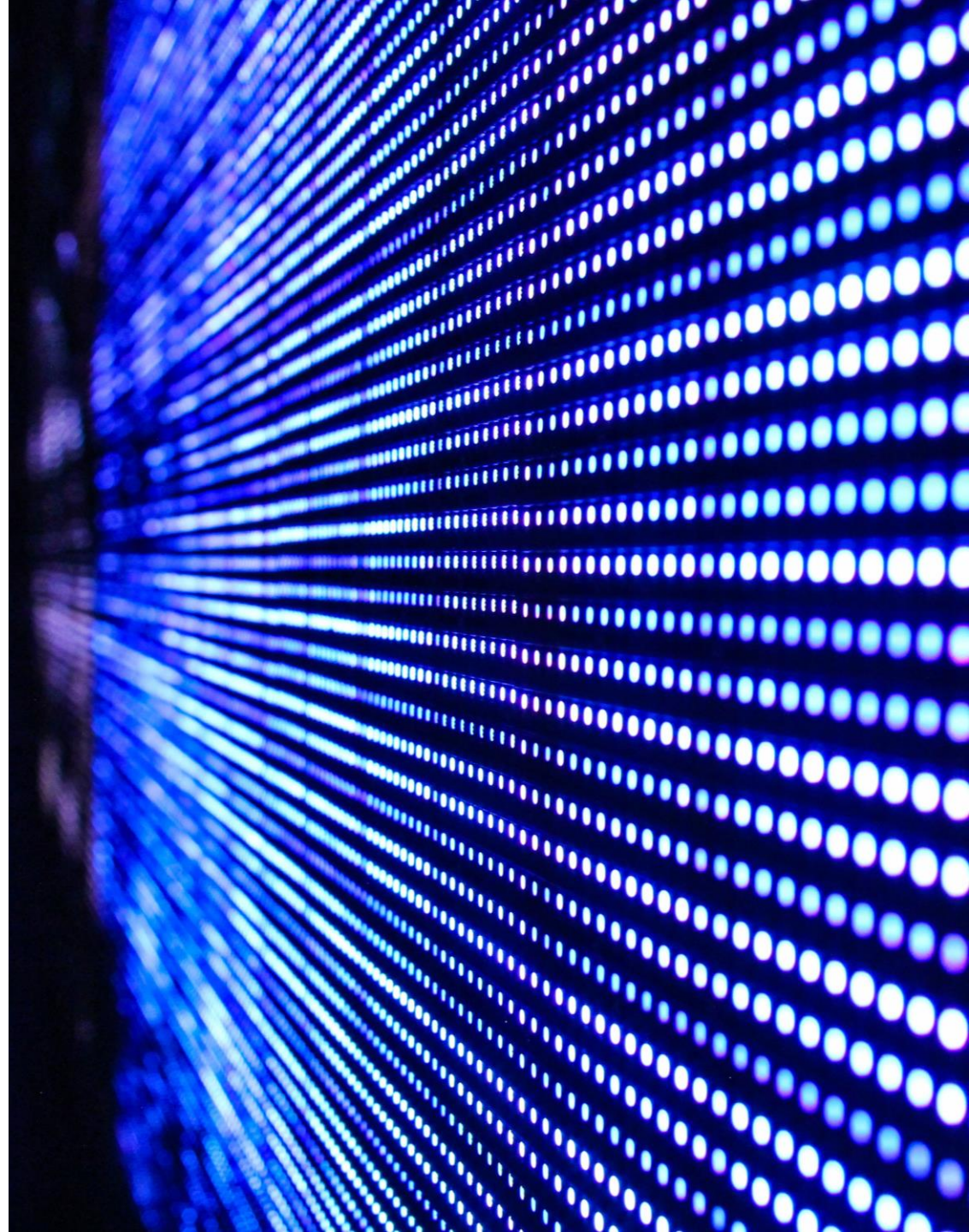
- Build at least beginning of the data pipeline up to storage
- Best if you can deploy software at the edge, to reduce volume, process, generate insights
- Tools: the usual MQTT/AMQT, AWS & Azure IoT
 - AWS IoT Analytics & AWS QuickSight
 - Your standard data and AI tools and services

VISIBLE (OBSERVABLE)

Open-source licenses

□ Ensure OS licenses compliance

- Save manifest of all license, and ship in the binary
 - `COPY_LIC_MANIFEST = "1"`
 - `COPY_LIC_DIRS = "1"`
 - `LICENSE_CREATE_PACKAGE = "1"`
- Archive source
 - `INHERIT += "archiver"`
`ARCHIVER_MODE[src] = "original" # OR`
`ARCHIVER_MODE[src] = "patched"`
`ARCHIVER_MODE[diff] = "1"`
- Some references:
 - Yocto's manual regarding compliance
 - OpenChain ISO 5230, Open Compliance Program
 - FOSSology license tracker
 - Various commercial tools



Devices should be...

4.2 Controllable

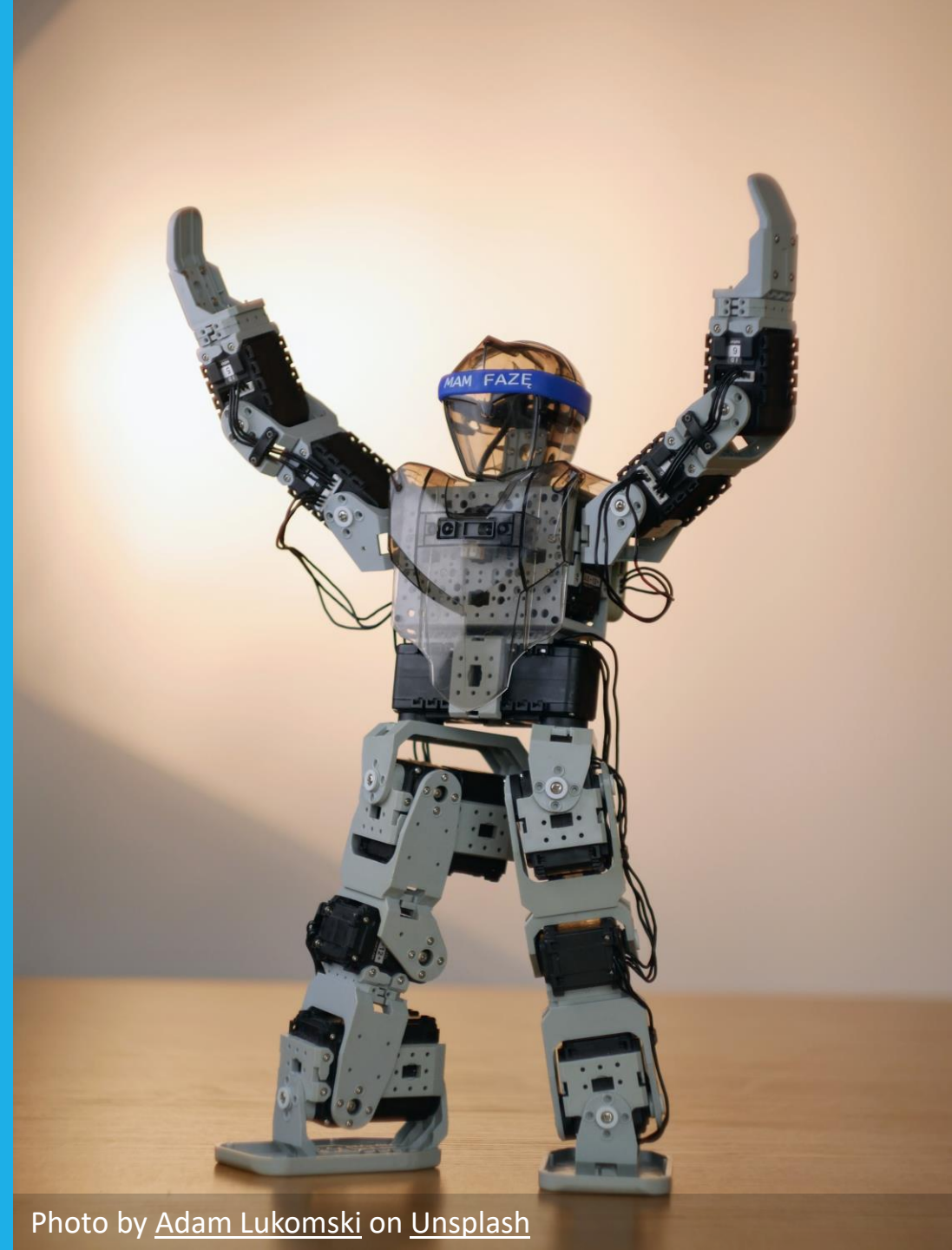


Photo by [Adam Lukomski](#) on [Unsplash](#)

CONTROLLABLE

Over-the-air updates

❑ Support OS updates

- OSTree, RAUC, Mender, swupdate
 - And their respective meta layers
 - Avoid non-atomic update like apt
- Kernel, packages, ...
- Sign your updates, look for delta updates

❑ (Nice to have) Support Application updates

- Different frequency, more flexibility & less bandwidth
- RAUC, Mender, OSTree, or managed deployment
 - Azure IoT Edge, AWS Greengrass, ...

❑ Provide OTA update online dashboards

- Hawkbit, Mender, ThingsBoard, Full Metal Update

❑ Provide a fallback mechanism

- What happens the OS, or update mechanism is KO?
- Recovery initramfs, A/B, golden/base image
- Or a combination of those
- **Test and re-test**

CONTROLLABLE

Remote operations

❑ Support remote device configuration

- Device twins... again!
- Cloud “desired” configuration, stored in same JSON
 - Desired network config, logging mode, CPU throttling (hot device), ...
- Received whenever connected, and persistent

❑ (Nice to have) Support arbitrary operations

- Provide a quick flexible way to support any operation
 - Specific repair job, test or experiment new features
- Running jobs and applications from container
 - Azure IoT Edge, AWS IoT jobs, AWS IoT Greengrass
 - Custom mechanism



CONTROLLABLE

Manual control and debugging

❑ Provide remote manual access

- Is it acceptable to ship the device back?
- Remote: Reverse SSH, OpenVPN/IPsec/wireguard, ngrok
- Local: GUI, secured USB drive script, ...
- Enabled on demand, for a limited duration
 - Update firewall rules, authorized devices, ...
 - or reboot in maintenance mode
 - After a secure call from your cloud platform and/or physical interaction

❑ (Nice to have) Support remote troubleshooting & debugging

- Include production loggers/tracers: LTTng
- Generate and save debug symbols
 - `IMAGE_GEN_DEBUGFS = "1"`
 - Install or run 'gdb-server' on-demand

Devices should be...

5. Reusable & Future proof



Photo by [Ravin Rau](#) on [Unsplash](#)

REUSABLE & FUTURE PROOF

Yocto and reusing OS-level work

❑ Limit “hacky” customizations

- Limit bbappends, and extensive patching

❑ Prefer flexible and reusable config

- Separate layers to allow reuse
 - Hardware, software, platform-specific features
- WIC: custom wks and partitions
 - More tools to interact, customize and introspect
- Device tree includes and overloads
- Yocto LTS

❑ Prepare by upgrading Yocto

- Once you know which is the ideal Yocto version
- Otherwise very hard to reuse meta/recipes/classes across different Yocto versions

❑ (Nice to have) Contribute layers, recipes, classes

- Generic layers, recipes, classes
 - Chances are others need it, and will help maintain them
 - Submit it to <https://layers.openembedded.org/>
- Yocto features & issues
 - create-pull-request, send-pull-request
 - bugzilla.yoctoproject.org

REUSABLE & FUTURE PROOF

Application-level

❑ Abstract device & OS specificities

- App software architecture
- Rely on standard file location and mechanics
- Makes development easier

❑ Ensure application(s) modularity

- Core logic, connectivity, UI, GUI, storage
- At a component level minimum:
 - Source components, plugins, or services
- Future: headless, gateway, split in 2 devices

❑ (Nice to have) Self-contained application

- Avoid dependencies, conflicts
- Containers, snapd/flatpak, self-contained binaries (Golang, Rust, ...)
- Easier to reuse, but larger

❑ Use standard tools and protocols

- Prefer what the industry uses (most of the time)
 - Conferences, blogs, Yocto mailing list, Gartner, ...
 - yocto@lists.yoctoproject.org
- When applicable, prefer standards for interop.
 - Matter, BLE profiles, ...

THE 5 SUCCESS FACTORS TO A PRODUCTION-READY DEVICE

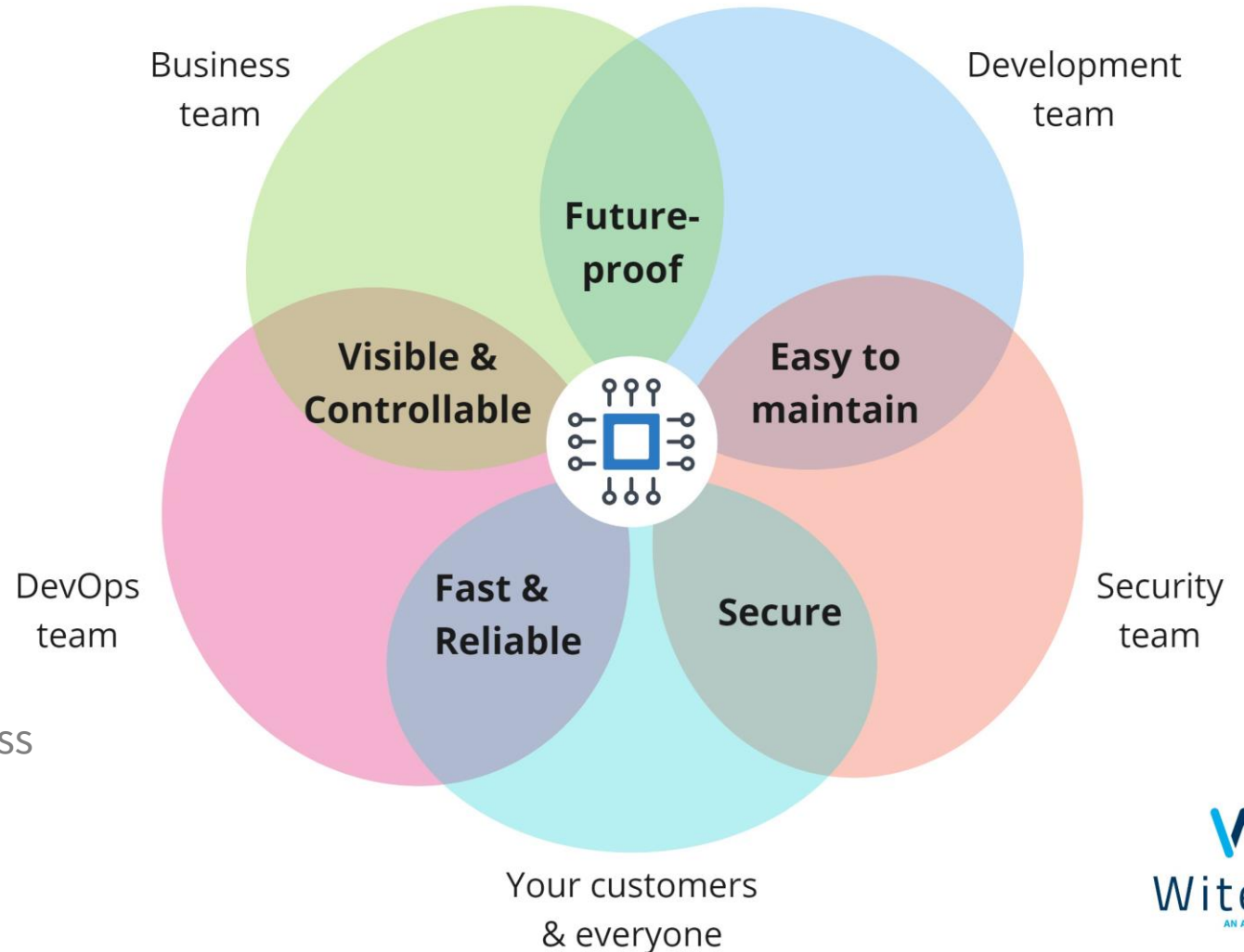
Summary

Use as a **checklist** before release

- The sooner the better!
- What would you add to this checklist?

The most important success factor: **you**

- Likely not all in your job description
- Internally: review, ask, and suggest features
- Learning opportunity, improve product readiness



Chat with us,
at our physical and virtual
booth

www.witekio.com



Adrien Leravat
Senior Software Architect
aleravat@witekio.com

Sid Nagendran
US Business Development Mgr.
sid@witekio.com

Witekio USA
3150 Richards Rd Suite 210
Bellevue, WA, 98005, USA
Phone : + 1 425-749- 4335